TriSP: Tri-Signal Structured Pruning for Large Language Models

Anonymous authors
Paper under double-blind review

Abstract

Large language models (LLMs) achieve strong performance across diverse tasks but their deployment is constrained by the memory and compute cost of their parameters. Structured pruning addresses this by removing entire structures such as attention heads and Multi-Layer Perceptron (MLP) neurons to produce smaller dense models that run efficiently on standard hardware. However, existing methods rely on either gradient-based importance estimation, which is memory-prohibitive, or activation-based statistical proxies, which do not directly measure the effect of removal on the loss. Furthermore, the interaction between the importance criterion and the post-pruning recovery strategy has not been systematically studied. We propose TriSP (Tri-Signal Structured Pruning), an importance metric that combines weight magnitude scaled by activation norm with first-order gradient sensitivity via a geometric mean, producing a channel-level score that captures both structural and loss-sensitivity signals. Combined with adaptive per-layer budget allocation and low-rank adaptation (LoRA) recovery, TriSP achieves the lowest perplexity and highest zero-shot accuracy across all tested configurations, reaching 6.80 WikiText-2 perplexity at 20% pruning on LLaMA-7B. Inference throughput improves by 82% at 50% pruning, while still maintaining competitive performance.

1 Introduction

Large language models (LLMs) based on the transformer architecture (Ashish, 2017) have become the dominant paradigm for natural language processing. Models such as GPT-4 (Achiam et al., 2023), LLaMA (Touvron et al., 2023), Mistral (Jiang et al., 2023), and DeepSeek (Bi et al., 2024) demonstrate strong performance across several tasks. However, their practical deployment is constrained by the computational and memory resources they require: a 7-billion parameter model in half-precision occupies approximately 13 GB of GPU memory and requires hundreds of billions of multiply-accumulate operations per forward pass (Zhu et al., 2024).

Structured pruning reduces this cost by removing entire architectural components: attention heads, Multi-Layer Perceptron (MLP) neurons, or channels, producing smaller dense models that benefit from standard GPU matrix operations. Unlike unstructured pruning, which zeros individual weights and requires specialized sparse kernels for speedup, structured pruning yields direct latency and memory reductions on commodity hardware. The central challenge is identifying which structures to remove with minimal quality degradation, which requires both an accurate importance criterion and an allocation strategy that distributes the pruning budget across layers in proportion to their redundancy.

Existing approaches address these requirements with complementary but incomplete solutions. FLAP (An et al., 2024) uses activation variance scaled by squared weight magnitude, a statistical proxy that is fast to compute but does not directly measure the effect of removal on the model's loss. LLM-Pruner (Ma et al., 2023) uses Taylor-based gradient importance with dependency graph analysis, which more directly estimates pruning impact but requires storing gradient tensors for all pretrained weights, making it memory-prohibitive for large models. Both methods apply post-pruning recovery (bias compensation for FLAP, LoRA fine-tuning

for LLM-Pruner) to restore quality, but no prior work has evaluated how the choice of recovery strategy affects the final quality.

Two research gaps motivate this work. First, gradient augmentation has consistently improved unstructured pruning: GBLM-Pruner (Das et al., 2023) adds a gradient term to Wanda's (Sun et al., 2023) activation-norm criterion, and Wanda++ (Yang et al., 2025) extends this with regional gradients. However, this approach has not been transferred to structured pruning, where gradient-based and activation-based methods remain separate. Second, FLAP's bias compensation is presented as sufficient to eliminate fine-tuning, yet quality degrades substantially at aggressive ratios, and no prior work has compared this static correction against LoRA recovery (Hu et al., 2022) under the same pruning criterion. This paper makes two contributions:

1. **TriSP (Tri-Signal Structured Pruning)**, an importance metric that combines activation-weighted magnitude with gradient sensitivity via a geometric mean, transferring gradient augmentation from unstructured to structured pruning.

2. A **systematic comparison** of bias compensation vs. LoRA recovery under the same pruning criterion, and comprehensive evaluation across five models.

The remainder of this paper is organized as follows: Section 2 reviews related work, Section 3 describes the proposed methodology, Section 4 presents the experimental setup and results, and Section 5 concludes the paper.

2 Related Work

Structured pruning for LLMs removes entire structural units (heads, neurons, channels) from a pretrained model to produce a smaller dense architecture that directly benefits from standard GPU kernels (He & Xiao, 2023). Methods differ along two axes: the importance criterion used to rank structures and whether post-pruning recovery is applied.

2.1 Gradient-Based Methods

First-order Taylor expansion provides a principled estimate of how much removing a structural unit affects the loss. LLM-Pruner (Ma et al., 2023) applies this criterion with approximate Hessian information, coupling parameters into structural groups (heads, MLP neurons) via a dependency graph and recovering quality with a lightweight LoRA adapter. However, it requires storing full gradient tensors for all pretrained weights, making it memory-intensive at scale. Two subsequent methods address this bottleneck by computing Taylor importance from the small LoRA adapter matrices instead: LoRAPrune (Zhang et al., 2024) enables pruning of LLaMA-65B on a single A100 GPU and applies iterative progressive pruning, while DyLoRA-Prune (Li & Guo, 2024) replaces fixed-rank LoRA with Dynamic Low-Rank Adaptation, learning representations across a range of ranks simultaneously so that lower ranks capture the most critical information.

An alternative line of work uses second-order information. SlimGPT (Ling et al., 2024) extends the Optimal Brain Surgeon (OBS) framework (Hassibi & Stork, 1992), previously applied to unstructured pruning (SparseGPT (Frantar & Alistarh, 2023)) and quantization (GPTQ (Frantar et al., 2022)), introducing Batched Greedy Pruning with grouped Cholesky decomposition for heads and dynamic group sizing for MLP channels. Even without fine-tuning, it reaches 52.23% zero-shot average at 50% on LLaMA-7B, outperforming LLM-Pruner with LoRA, while requiring only 7 GB of GPU memory at 20%.

2.2 Gradient-Free Methods

Activation-based criteria avoid gradient computation entirely. FLAP (An et al., 2024) scores channels by the sample variance of input activations weighted by squared weight-column norm, replacing low-fluctuation channels with a fixed bias correction that requires no gradients. Similarly, Bonsai (Kolawole et al., 2024) frames importance as a regression problem, fitting a linear model over randomly sampled sub-models to

estimate each module's contribution globally, enabling pruning on a single 48 GB A6000 where gradient-based methods require 80–640 GB. Probe Pruning (Le et al., 2025) extends the activation-based paradigm by making decisions online: a lightweight probe subset (5% of samples) runs ahead through a few layers, fusing resulting states with historical calibration states so that current batch characteristics dominate when they deviate from calibration, surpassing FLAP (61.0% vs. 60.6% commonsense accuracy at 40% on LLaMA-2-13B) with only 1.5% overhead in FLOPs.

A separate family of gradient-free methods operates at coarser granularity. SliceGPT (Ashkboos et al., 2024) exploits the computational invariance of RMSNorm transformers under orthogonal rotations, concentrating signal into leading PCA components and slicing off small-eigenvalue dimensions uniformly across the model. At the layer level, ShortGPT (Men et al., 2025) removes entire transformer layers based on Block Influence, observing that middle layers carry the most redundancy, while BlockPruner (Zhong et al., 2025) refines this by treating MHA and MLP blocks as separate pruning units and finding asymmetric redundancy: below 17% pruning MHA blocks are more dispensable, but beyond that threshold MHA becomes critical while MLP remains tolerant.

2.3 Gradient Augmentation in Unstructured Pruning

In unstructured pruning, Wanda (Sun et al., 2023) demonstrates that a simple activation-aware criterion $S_{ij} = |W_{ij}| \cdot \|X_j\|_2$ can match SparseGPT's accuracy without weight reconstruction. GBLM-Pruner (Das et al., 2023) extends this score with a gradient term, consistently outperforming both Wanda and SparseGPT on LLaMA-2-7B at 50% sparsity. Wanda++ (Yang et al., 2025) further reduces memory cost by restricting gradient computation to single decoder blocks and applies regional weight updates to minimize pruning-induced output discrepancy. However, this gradient augmentation has not been transferred to structured channel scoring.

In this work, we propose TriSP (Tri-Signal Structured Pruning), which bridges this gap by transferring gradient augmentation from unstructured to structured pruning. Existing structured methods fall on one side: gradient-based methods like LLM-Pruner directly estimate pruning impact but require storing full gradient tensors, while gradient-free methods like FLAP use activation statistics that are cheap but do not measure the effect of removal on the loss. TriSP combines both signals into a single channel-level score while accumulating gradients only for the two output projections per layer, reducing gradient storage compared to full Taylor expansion.

3 Methodology

The proposed method operates as a three-stage pipeline (Figure 1): importance estimation, structural pruning via adaptive layer-module (AL-AM) allocation, and an optional low-rank adaptation (LoRA) recovery.

3.1 Calibration

Importance estimation requires collecting activation statistics and gradients from a small calibration set. For each sample, we perform a full forward pass through the model to obtain logits, compute the per-sample cross-entropy loss $\mathcal{L}$ on next-token prediction:

$$\mathcal{L} = -\frac{1}{T-1} \sum_{t=1}^{T-1} \log P(x_{t+1} \mid x_1, \ldots, x_t) \tag{1}$$

where T is the sequence length and x_t is the token at position t. We then execute a backward pass to produce weight gradients. The absolute gradients are accumulated across all N calibration samples to form a per-weight average gradient magnitude:

$$\bar{G}_{l,ij} = \frac{1}{N} \sum_{t=1}^{N} \left| \frac{\partial \mathcal{L}^{(t)}}{\partial W_{l,ij}} \right| \tag{2}$$

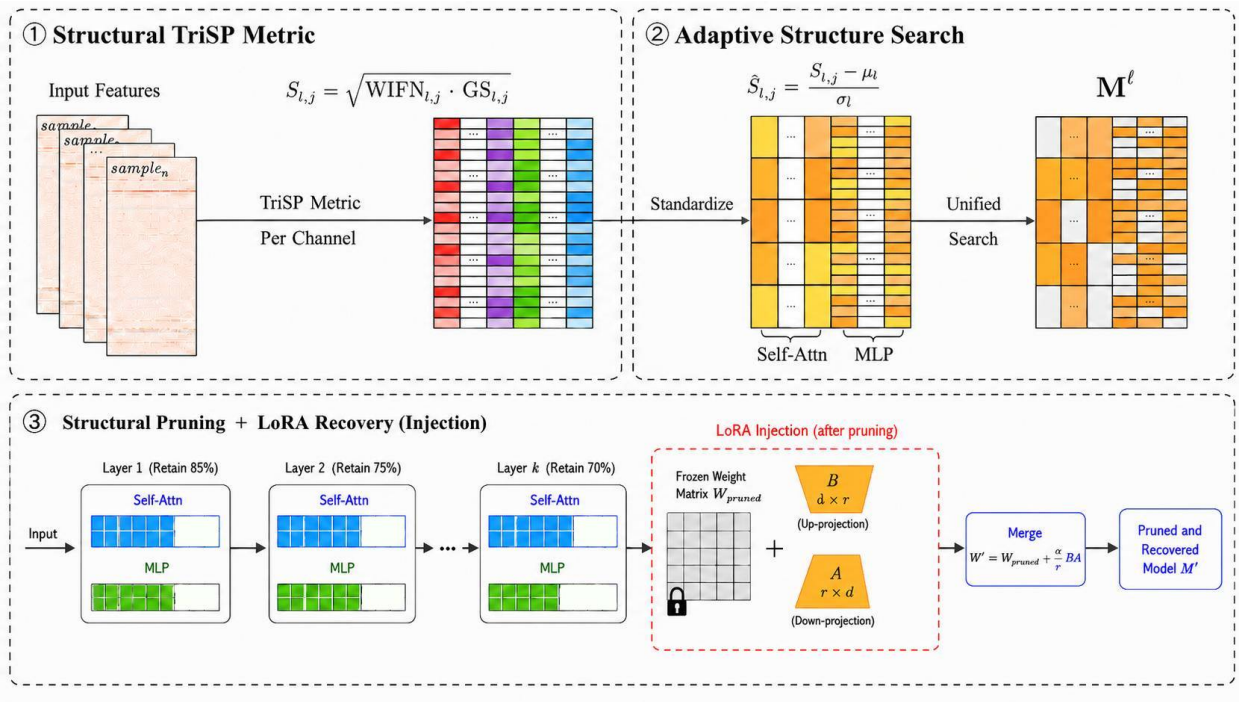

Figure 1: Overview of the proposed TriSP structured pruning pipeline. **(1)** Compute tri-signal importance scores $S_{l,j}$, the geometric mean of $\text{WIFN}_{l,j}$ and $\text{GS}_{l,j}$, combining weight magnitude, gradient sensitivity, and activation norm per channel from calibration data; **(2)** Standardize scores across layers via z-score normalization and apply a unified threshold; **(3)** Physically remove pruned heads and neurons with adaptive per-layer retention ratios, then inject LoRA adapters into the frozen pruned weights and merge after fine-tuning.

where $W_{l,ij}$ denotes the weight at row i, column j of a linear layer in transformer layer l.

During the forward pass, input activation statistics are collected at the output projection and down projection of each transformer layer as a running squared ℓ_2 norm:

$$a_{l,j}^{(t)} = \frac{n}{n+b}\, a_{l,j}^{(t-1)} + \frac{\|X_{l,j}^{(t)}\|_2^2}{n+b} \tag{3}$$

where $a_{l,j}^{(t)}$ is the accumulated squared activation norm for input channel j in layer l at step t, $X_{l,j}^{(t)}$ is the corresponding activation vector, n is the number of samples seen so far, and b is the current batch size. After all calibration batches, $a_{l,j}$ converges to the mean squared activation norm at channel j.

Although TriSP requires an additional backward pass compared to FLAP, this roughly doubles the per-sample compute only during calibration, which is a one-time cost that completes in minutes on a single GPU, and the improved pruning quality justifies the overhead.

3.2 Tri-Signal Structured Metric

We propose a tri-signal importance metric that combines a structural signal from FLAP (An et al., 2024) with a gradient-based sensitivity signal. For a linear layer with weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, we define two per-channel components. The first is the *Weighted Input Feature Norm* (WIFN), which captures weight

magnitude scaled by activation norm:

$$\mathrm{WIFN}_{l,j} = \frac{1}{d_{\text{out}}} \sum_{i=1}^{d_{\text{out}}} |W_{l,ij}| \cdot \sqrt{a_{l,j} + \epsilon} \tag{4}$$

where $W_{l,ij}$ is the weight connecting input channel j to output channel i in layer l, d_{out} is the number of output channels, $a_{l,j}$ is the accumulated squared activation norm from Equation 3, and $\epsilon = 10^{-6}$.

The second is the *Gradient Sensitivity* (GS) term, which captures how much the loss depends on each channel, again scaled by activation norm:

$$\mathrm{GS}_{l,j} = \left(\frac{1}{d_{\text{out}}} \sum_{i=1}^{d_{\text{out}}} |W_{l,ij}| \cdot \bar{G}_{l,ij} \right) \cdot \sqrt{a_{l,j} + \epsilon} \tag{5}$$

where $\bar{G}_{l,ij}$ is the averaged absolute gradient from Equation 2. The final importance score is the geometric mean of these two components:

$$S_{l,j} = \sqrt{\mathrm{WIFN}_{l,j} \cdot \mathrm{GS}_{l,j}} \tag{6}$$

This formulation captures three complementary signals:

- **Weight magnitude** $|W_{l,ij}|$: channels connected through large weights carry more information.
- **Gradient magnitude** $\bar{G}_{l,ij}$: channels with high gradient magnitude indicate higher importance for model performance.
- **Input activation norm** $\sqrt{a_{l,j}}$: channels that receive large activations during calibration contribute more to the layer output, consistent with the observation by Sun et al. (2023) that outlier features in LLMs carry disproportionate importance.

The geometric mean prevents the gradient term from dominating the score when gradients are noisy or have high variance across layers.

3.3 Adaptive Layer-Module Allocation

Following FLAP (An et al., 2024), we adopt an adaptive allocation strategy that distributes pruning budgets across layers and module types based on measured redundancy.

Cross-layer standardization. The raw TriSP scores differ in magnitude across layers because deeper layers tend to have different activation scales and gradient norms than shallow ones. Applying a single global threshold to raw scores would prune too aggressively in layers with small scores and too conservatively in layers with large scores. To make scores comparable across layers, we apply per-layer z-score standardization:

$$\hat{S}_{l,j} = \frac{S_{l,j} - \mu_l}{\sigma_l} \tag{7}$$

where $S_{l,j}$ is the TriSP importance score of channel j in layer l, and μ_l and σ_l are the mean and standard deviation of all channel scores within layer l.

Attention head scoring. For multi-head attention, the per-channel TriSP scores of the output projection are grouped by head. Each head spans d_h contiguous channels, and the head-level importance is the mean of its channel scores:

$$H_{l,k} = \frac{1}{d_h} \sum_{j \in \mathrm{head}_k} \hat{S}_{l,j} \tag{8}$$

where $H_{l,k}$ is the importance score of head k in layer l, d_h is the head dimension, and $\hat{S}_{l,j}$ are the z-score normalized channel scores from Equation 7.

Multi-Layer Perceptron neuron scoring. Each MLP neuron corresponds to one intermediate dimension, shared across the gate, up, and down projections. The standardized TriSP scores of the down projection's input channels are used directly as per-neuron importance scores.

Global threshold with compression weighting. All head scores and neuron scores across all layers are concatenated into a single vector and sorted in descending order. Because removing one attention head eliminates $4 \times d_h \times d_{\text{hidden}}$ parameters (across four projections: W_Q, W_K, W_V, and W_O), while removing one MLP neuron eliminates only $3 \times d_{\text{hidden}}$ parameters (across three projections: W_{gate}, W_{up}, and W_{down}), pruning a head removes far more parameters than pruning a neuron. To account for this, each component is assigned a *compression weight* c_i that reflects its relative parameter cost: each attention head receives $c_i = 4d_h/3$, while each MLP neuron receives $c_i = 1$.

The global threshold is determined in two steps. First, the optimal cutoff position k^* is found by scanning the sorted list and selecting the position where the cumulative compression weight is closest to the total compression weight that should be retained:

$$k^* = \arg\min_k \left| \sum_{m=1}^{k} c_{\pi(m)} - (1-p) \sum_i c_i \right| \tag{9}$$

where π is the permutation that sorts all component scores in descending order, $c_{\pi(m)}$ is the compression weight of the component at sorted position m, p is the target pruning ratio, and $(1-p)\sum_i c_i$ is the total compression weight. Then, the threshold τ is set to the score at that position:

$$\tau = S_{\pi(k^*)} \tag{10}$$

All components with $S_i > \tau$ are retained; the rest are pruned.

3.4 Structural Pruning

Once the binary masks are determined, we compress the model by physically removing the pruned dimensions from the weight matrices, producing a smaller dense model.

The rows of the query projection W_Q, key projection W_K, and value projection W_V corresponding to pruned heads are removed, and the corresponding columns of the output projection W_O are removed. For the MLP, we remove the rows of the up projection W_{up} and gate projection W_{gate} corresponding to pruned neurons, and the corresponding columns of the down projection W_{down}. The number of retained MLP neurons per layer is rounded down to the nearest multiple of 64 to ensure efficient GPU memory access and matrix computation.

In the original FLAP formulation, a fixed bias vector compensates for the removed channels by adding the mean activation of pruned inputs to the layer output: $b_{\text{comp}} = W_{\text{out}}^{\top} \cdot \bar{x}_{\text{pruned}}$, where W_{out} is the weight matrix of the output projection and $\bar{x}_{\text{pruned}}$ is the mean activation vector of the pruned channels computed over the calibration set. This is a rank-zero approximation that captures only the mean and ignores per-sample variation.

We deliberately omit this bias term, as the LoRA adapters applied in the recovery stage learn a richer, input-dependent correction that subsumes what the fixed bias provides (see Figure 2).

3.5 Low-Rank Adaptation Recovery

Structural pruning introduces a quality-accuracy trade-off. To recover performance efficiently, we employ Low-Rank Adaptation (LoRA) (Hu et al., 2022) as a post-pruning recovery step. LoRA injects low-rank adapter matrices into the pruned model's attention projections (W_Q, W_K, W_V, and W_O) of every transformer layer. Only the adapter parameters are trained while all pretrained weights remain frozen, making the recovery lightweight. After training, adapters are merged back into the pruned weights.

Algorithm 1 TriSP: Tri-Signal Structured Pruning with LoRA Recovery

Require: Model $\mathcal{M}$ with L transformer layers, calibration set $\mathcal{D}_{\text{cal}}$ (N=512 samples), pruning ratio p, LoRA rank r, recovery dataset $\mathcal{D}_{\text{rec}}$
Ensure: Pruned and recovered model $\mathcal{M}'$
Stage 1: Importance Estimation
for each layer $l = 1, \ldots, L$ **do**
  Initialize $a_{l,j} \leftarrow 0$, $\bar{G}_{l,ij} \leftarrow 0$; collect input activations at $W_O^{(l)}$ and $W_{\text{down}}^{(l)}$
end for
for each sample $(x, y) \in \mathcal{D}_{\text{cal}}$ **do**
  Forward pass: $\hat{y} \leftarrow \mathcal{M}(x)$; compute $\mathcal{L}$ (Eq. 1); backward pass
  for each layer l **do**
    Accumulate $\bar{G}_{l,ij}$ (Eq. 2); update $a_{l,j}$ (Eq. 3)
  end for
end for
for each layer l **do**
  Compute $\text{WIFN}_{l,j}$ (Eq. 4), $\text{GS}_{l,j}$ (Eq. 5), TriSP score $S_{l,j}$ (Eq. 6)
end for
Stage 2: Adaptive Layer-Module (AL-AM)
for each layer l **do**
  Standardize scores $\hat{S}_{l,j}$ via z-score normalization (Eq. 7)
end for
Aggregate per-head importance $H_{l,k}$ (Eq. 8);
Sort the scores in descending order
Assign compression weights: $c_i \leftarrow 4d_h/3$ for heads, $c_i \leftarrow 1$ for neurons
Find global threshold τ for target ratio p (Eq. 9); generate masks $M_i \leftarrow \mathbf{1}[\text{score}_i > \tau]$
Stage 3: Structural Pruning and Recovery
for each layer l **do**
  Remove pruned head rows from W_Q, W_K, W_V and columns from W_O; remove pruned neuron rows from $W_{\text{up}}, W_{\text{gate}}$ and columns from W_{down};
end for
Freeze all weights; inject LoRA adapters into W_Q, W_K, W_V, W_O
Fine-tune adapters on $\mathcal{D}_{\text{rec}}$ for 2 epochs; merge adapters into pruned weights
return $\mathcal{M}'$

Table 1: Architectural configurations of all models used in this work.

Model	L	d	H_Q	H_{KV}	d_{ff}	$\lvert V\rvert$	**Context**
LLaMA-2-7B	32	4096	32	32	11 008	32 000	4 096
Vicuna-7B	32	4096	32	32	11 008	32 000	4 096
DeepSeek-7B	30	4096	32	32	11 008	102 400	4 096
Mistral-7B	32	4096	32	8	14 336	32 000	32 768

L: layers, d: hidden dim., H_Q: query heads, H_{KV}: KV heads, d_{ff}: intermediate dim., $|V|$: vocab size.

4 Experiments and Results

4.1 Experimental Setup

Models. The primary pruning experiments are conducted on LLaMA-2-7B (Touvron et al., 2023) and Vicuna-7B (Chiang et al., 2023) at pruning ratios of 20%, 30%, and 50%. To assess generalization across architectures, the same pipeline is additionally evaluated on DeepSeek-7B (Bi et al., 2024) and Mistral-7B (Jiang et al., 2023). Table 1 summarizes the architectural configurations of all models.

Baselines. The proposed TriSP pruning pipeline is compared against FLAP (An et al., 2024) (with and without bias compensation) and LLM-Pruner (Ma et al., 2023) (with and without LoRA recovery). All methods use the same calibration set (512 WikiText-2 samples) for fair comparison.

Metrics. We evaluate four dimensions: (i) language modeling quality via WikiText-2 test and Penn Treebank perplexity (PTB); (ii) downstream task accuracy via zero-shot evaluation on six commonsense benchmarks: PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy and ARC-Challenge (Clark et al., 2018), and OBQA (Mihaylov et al., 2018); (iii) computational

Table 2: WikiText-2 perplexity of pruning methods across two models and three pruning ratios. Best results in **bold**.

Ratio	Method	Recovery	LLaMA-7B	Vicuna-7B
0%	Dense	–	5.47	6.78
20%	FLAP	w/o tune	7.52	9.50
		bias	7.11	9.04
	TriSP	w/o tune	7.06	8.81
		LoRA	**6.80**	**7.77**
30%	FLAP	w/o tune	9.87	12.72
		bias	8.61	11.38
	TriSP	w/o tune	9.03	10.95
		LoRA	**8.05**	**9.10**
50%	FLAP	w/o tune	29.43	36.86
		bias	16.03	22.45
	TriSP	w/o tune	27.63	31.67
		LoRA	**13.44**	**14.82**

cost via MACs; and (iv) inference efficiency via forward-pass latency and throughput, measured over 100 timed runs at batch size 32 with sequence length 64 on a single NVIDIA H100 80GB GPU.

LoRA recovery. LoRA adapters with rank $r = 8$ and scaling factor $\alpha = 16$ are applied to the four attention projections of every transformer layer. A dropout rate of 0.05 is applied to the LoRA matrices. Recovery fine-tuning is performed on 25,000 samples of the Alpaca instruction-tuning dataset (Taori et al., 2023) for 2 epochs, with a learning rate of 10^{-4}, cosine schedule. Extending LoRA to MLP projections is left for future work, as attention-only recovery was sufficient to achieve the best results at all tested ratios.

4.2 Language Modeling Perplexity

Table 2 reports WikiText-2 perplexity across two models and three pruning ratios. At all ratios and across both models, TriSP with LoRA recovery achieves the lowest perplexity. On LLaMA-7B, TriSP + LoRA reaches 6.80 at 20%, 8.05 at 30%, and 13.44 at 50%, consistently outperforming both FLAP without recovery and FLAP with bias compensation. On Vicuna-7B at 50%, TriSP + LoRA achieves 14.82 compared to 22.45 for FLAP + bias, a 34% relative improvement.

4.3 Zero-Shot Downstream Tasks

Table 3 evaluates zero-shot accuracy on six commonsense reasoning benchmarks for LLaMA-2-7B. At 20% pruning, TriSP + LoRA achieves the highest average accuracy (60.98%), outperforming LLM-Pruner + LoRA (58.78%), FLAP + bias (56.34%), and FLAP without bias (54.51%). TriSP + LoRA leads on all six benchmarks, with particularly strong margins on HellaSwag (68.20 vs. 64.57 for LLM-Pruner) and Wino-Grande (66.85 vs. 64.56 for FLAP).

At 50% pruning, all methods degrade substantially, but TriSP + LoRA retains the best average performance (48.18%), with notable margins over all baselines, including LLM-Pruner + LoRA on PIQA (+5.06), ARC-e (+11.58), and HellaSwag (+10.61).

Table 3: Zero-shot commonsense reasoning accuracy (%) on LLaMA-2-7B at 20% and 50% pruning. Best results in **bold**.

Ratio	Method	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg.
0%	LLaMA-7B	78.13	74.06	69.38	75.46	43.09	42.60	63.79
20%	LLM-Pruner w/o tune	74.43	61.46	59.75	68.43	34.64	39.40	56.35
	LLM-Pruner w/ LoRA	75.63	64.57	61.17	72.52	38.57	40.20	58.78
	FLAP w/o bias	72.80	61.51	64.01	60.31	29.61	38.80	54.51
	FLAP w/ bias	73.72	63.14	64.56	64.44	32.17	40.00	56.34
	TriSP w/o tune	75.90	67.97	65.67	70.54	36.43	39.60	59.35
	TriSP w/ LoRA	**76.77**	**68.20**	**66.85**	**73.02**	**40.02**	**41.00**	**60.98**
50%	LLM-Pruner w/o tune	53.48	27.03	48.54	30.18	20.65	24.80	34.11
	LLM-Pruner w/ LoRA	62.40	37.56	51.22	44.44	22.10	28.40	41.02
	FLAP w/o bias	60.01	42.56	50.75	36.78	22.95	32.00	40.84
	FLAP w/ bias	60.94	40.62	53.04	43.98	22.70	33.60	42.48
	TriSP w/o tune	61.48	42.42	51.62	45.92	25.09	32.20	43.12
	TriSP w/ LoRA	**67.46**	**48.17**	**53.04**	**56.02**	**29.78**	**34.60**	**48.18**

4.4 Generalization to Other Architectures

To validate that TriSP generalizes beyond the LLaMA-2 family, we evaluate on DeepSeek-7B and Mistral-7B, which differ in several architectural dimensions detailed in Table 1.

DeepSeek-7B. Table 4 reports perplexity and zero-shot accuracy at 20% and 30% pruning. TriSP + LoRA achieves the best WikiText-2 perplexity at both ratios (8.27 at 20%, 9.82 at 30%) and the highest average zero-shot accuracy (61.95% at 20%, 58.62% at 30%). At 20%, TriSP + LoRA improves over FLAP + bias by 2.62 points in average accuracy and over LLM-Pruner + LoRA by 2.93 points. The gap widens at 30%, where TriSP leads FLAP + bias by 3.26 points.

Mistral-7B. Table 5 shows results on Mistral-7B. TriSP + LoRA achieves the best average accuracy at both ratios (62.64% at 20%, 58.02% at 30%). At 30%, TriSP outperforms FLAP + bias by 8.31 points (58.02 vs. 49.71) and LLM-Pruner + LoRA by 3.78 points. Notably, TriSP without LoRA achieves the lowest WikiText-2 perplexity at 20% (6.15 vs. 6.35 for TriSP + LoRA), however, LoRA still improves average downstream accuracy (62.64 vs. 59.51).

4.5 Inference Efficiency

Table 6 reports inference statistics for LLaMA-2-7B models. At 20% pruning, all three methods achieve comparable parameter counts (∼5.4B) and MACs (∼340G). The key differentiator is latency: TriSP achieves an 18% latency reduction (40.94 ms vs. 49.83 ms for dense) and a 22% throughput improvement over the dense baseline (50.0k tokens/s vs. 41.1k), while both LLM-Pruner (50.47 ms) and FLAP (49.40 ms) show weak speedup. This difference is likely because TriSP rounds MLP dimensions to multiples of 64, ensuring efficient GPU memory access and matrix computation, while LLM-Pruner and FLAP produce dimensions that do not align with the hardware execution granularity.

At 50%, LLM-Pruner achieves the lowest latency (26.11 ms, ↓48%), with TriSP close behind (27.46 ms, ↓45%), while FLAP lags at 34.02 ms (↓32%). TriSP delivers 74.6k tokens/s, an 82% throughput improvement over the dense baseline, compared to 78.4k (↑91%) for LLM-Pruner and 60.2k (↑46%) for FLAP. Memory savings scale proportionally with parameter reduction, dropping from 12,885 MiB to ∼6,700 MiB at 50%.

Table 4: Perplexity and zero-shot accuracy (%) of pruning methods for DeepSeek-7B.

Ratio	Method	WikiText-2 ↓	PTB ↓	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg. ↑
0%	Dense	6.84	9.27	78.51	73.85	70.48	74.58	42.32	45.20	64.16
20%	LLM-Pruner w/ LoRA	9.79	14.96	77.20	66.12	63.14	70.41	36.26	41.00	59.02
	FLAP w/ bias	8.30	11.58	76.39	66.58	**65.51**	67.89	35.84	43.80	59.34
	TriSP w/o tune	8.53	12.19	77.31	**70.32**	64.40	68.98	37.63	43.60	60.37
	TriSP w/ LoRA	**8.27**	**11.92**	**77.80**	69.87	65.04	**72.60**	**41.81**	**44.60**	**61.95**
30%	LLM-Pruner w/ LoRA	15.89	38.27	74.32	58.10	56.99	64.02	31.74	38.40	53.93
	FLAP w/ bias	10.00	**13.98**	73.39	59.85	**63.06**	62.04	33.45	40.40	55.37
	TriSP w/o tune	10.68	15.54	75.19	**65.58**	60.54	64.86	34.98	42.60	57.29
	TriSP w/ LoRA	**9.82**	14.34	**75.68**	65.43	62.04	**68.86**	**36.52**	**43.20**	**58.62**

Table 5: Perplexity and zero-shot accuracy (%) of pruning methods for Mistral-7B.

Ratio	Method	WikiText-2 ↓	PTB ↓	PIQA	HellaSwag	WinoGrande	ARC-e	ARC-c	OBQA	Avg. ↑
0%	Dense	5.25	26.96	80.25	78.75	74.66	80.22	49.83	46.60	68.39
20%	LLM-Pruner w/ LoRA	8.14	30.81	75.68	66.97	63.38	71.17	41.47	41.40	60.01
	FLAP w/ bias	6.46	42.14	69.10	52.22	66.54	66.71	33.79	40.80	54.86
	TriSP w/o tune	**6.15**	34.39	74.32	61.94	**69.06**	70.83	39.51	41.40	59.51
	TriSP w/ LoRA	6.35	**32.83**	**76.88**	**67.49**	67.56	**74.87**	**45.82**	**43.20**	**62.64**
30%	LLM-Pruner w/ LoRA	11.72	52.62	**72.58**	59.20	58.09	63.51	33.87	38.20	54.24
	FLAP w/ bias	7.78	57.22	65.56	44.78	61.64	59.85	28.84	37.60	49.71
	TriSP w/o tune	7.26	43.31	67.79	51.97	64.25	65.78	32.94	39.20	53.66
	TriSP w/ LoRA	**7.04**	**38.27**	72.47	**61.29**	**65.67**	**69.53**	**38.14**	**41.00**	**58.02**

4.6 Ablation Studies

We systematically examine three components of the TriSP pipeline: the recovery strategy, the importance metric, and the allocation strategy.

4.6.1 Effect of Recovery Strategy

Figure 2 compares three recovery strategies for TriSP pruning on Vicuna-7B: no recovery, bias compensation, and LoRA fine-tuning. LoRA recovery performs best across all pruning ratios, reducing perplexity from 8.81 to 7.77 at 20%, from 10.95 to 9.10 at 30%, from 15.83 to 10.96 at 40%, and from 31.67 to 14.82 at 50%. Bias compensation provides intermediate improvements at moderate ratios (e.g., 9.65 at 30% and 13.02 at 40%), but its advantage over no recovery diminishes relative to LoRA as compression increases. At 50%, the gap between LoRA and bias widens sharply (14.82 vs. 20.79). At 40%, LoRA reduces perplexity by 30.7% relative to no recovery, while bias compensation achieves only 17.8%. This confirms that LoRA's input-dependent, rank-r correction subsumes and exceeds what a rank-zero fixed bias term can provide.

4.6.2 Importance Metric Ablation

Table 7 isolates the contribution of each component in the TriSP metric by evaluating WIFN alone, GS alone, and the combined TriSP score on Vicuna-7B. At all pruning ratios, the combined TriSP metric outperforms either component in isolation: at $p = 0.4$, TriSP achieves 15.83 compared to 16.12 for WIFN and 17.14 for GS. The gap for the GS component grows from 0.10 at $p = 0.2$ to 1.31 at $p = 0.4$, suggesting that gradient sensitivity alone overprunes certain channels that WIFN's activation-weighted magnitude signal protects.

4.6.3 Allocation Strategy Ablation

Figure 3 compares uniform layer-module allocation (UL-UM), which assigns the same pruning ratio to every layer and module type, against adaptive layer-module allocation (AL-AM) on LLaMA-2-7B. At mild pruning ($p \leq 0.3$), the gap is negligible ($\Delta \leq 0.34$ perplexity points). Beyond 30%, the benefit grows sharply: at $p = 0.4$, AL-AM reduces perplexity by 8.27 points (13.84 vs. 22.11), and at $p = 0.5$, the gap reaches 25.83

Table 6: Inference statistics of compressed LLaMA-2-7B. Measured with sequence length 64, batch size 32 on a single NVIDIA H100.

Method	Ratio	Params	MACs	Memory	Latency (ms)	Tokens/s
LLaMA-2-7B	0%	6.74B	422.85G	12885 MiB	49.83	41.1k
LLM-Pruner	20%	5.47B	341.67G	10433 MiB	50.47 (≈0%)	40.6k
FLAP	20%	5.44B	340.06G	10424 MiB	49.40 (≈0%)	41.5k
TriSP (Ours)	20%	5.43B	339.35G	10462 MiB	40.94 (↓18%)	50.0k
LLM-Pruner	50%	3.50B	215.62G	6677 MiB	<u>26.11</u> (↓48%)	78.4k
FLAP	50%	3.50B	215.67G	6726 MiB	34.02 (↓32%)	60.2k
TriSP (Ours)	50%	3.49B	214.82G	6751 MiB	27.46 (↓45%)	74.6k

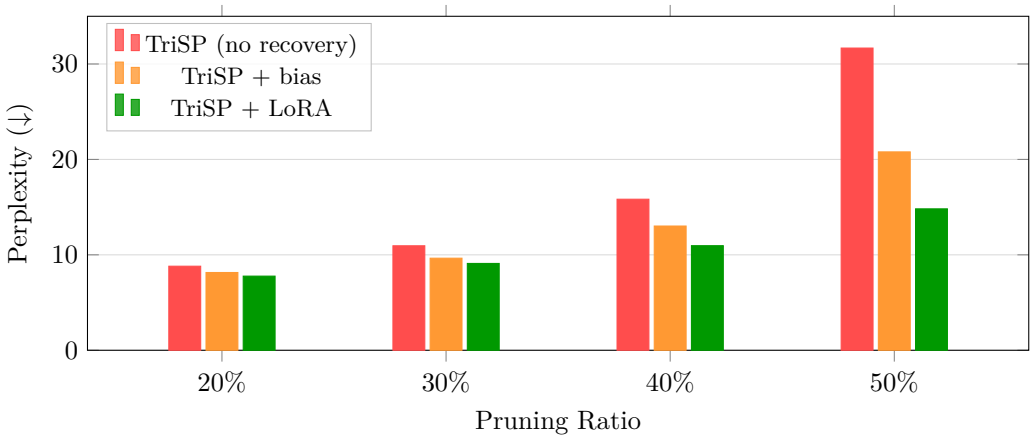

Figure 2: Vicuna-7B: WikiText-2 perplexity under TriSP pruning with different recovery strategies.

points (27.63 vs. 53.46). This demonstrates that adaptive budget redistribution becomes critical at aggressive pruning, where uniform allocation causes catastrophic over-pruning in sensitive layers.

4.7 Discussion

Why TriSP outperforms FLAP. FLAP uses activation variance as a statistical proxy for importance; TriSP includes $|W_{ij}| \cdot \bar{G}_{ij}$ from the Taylor expansion, which directly measures first-order loss sensitivity. This explains why TriSP without recovery already outperforms FLAP without recovery at all ratios: the metric itself makes better pruning decisions, not just the recovery strategy.

Why TriSP outperforms LLM-Pruner. LLM-Pruner models structural dependencies through dependency graphs, while TriSP treats channels independently. Despite this, TriSP without recovery achieves higher average zero-shot accuracy than LLM-Pruner without recovery (59.35 vs. 56.35 at 20%) and even surpasses LLM-Pruner with LoRA (58.78), indicating that the TriSP importance criterion itself makes better pruning decisions before any recovery is applied. LLM-Pruner's dependency graph constrains which structures can be removed together, which may force the removal of important channels that happen to be coupled with unimportant ones.

Recovery adequacy. At mild pruning (20%), bias compensation and LoRA yield similar improvements, because few channels are removed and the rank-zero mean approximation is sufficient. At aggressive pruning (40–50%), LoRA's rank-r input-dependent correction vastly outperforms the fixed bias, because a fixed offset (bias) cannot capture the input-dependent variation lost when many channels are removed.

Practical deployment recommendations. At 20–30% pruning, perplexity increases by 1–3 points and zero-shot accuracy drops by 2–6 points, with 14–24% latency reduction. This operating range is suitable for

Table 7: Metric ablation on Vicuna-7B: WikiText-2 perplexity using individual importance signals vs. the full TriSP metric.

Metric	$p = 0.2$	$p = 0.3$	$p = 0.4$
WIFN only	8.92	11.26	16.12
GS only	8.91	12.17	17.14
TriSP (ours)	**8.81**	**10.95**	**15.83**

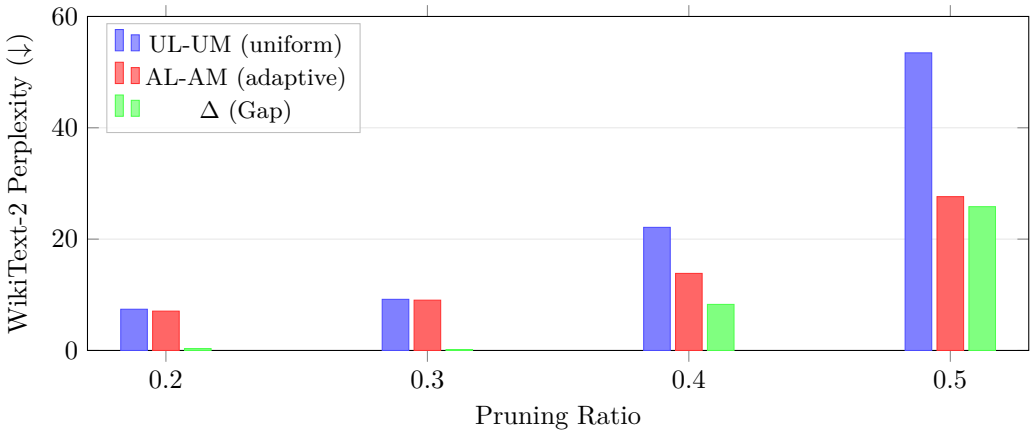

Figure 3: WikiText-2 perplexity: uniform (UL-UM) vs. adaptive (AL-AM) allocation on LLaMA-2-7B.

most applications where modest quality loss is acceptable for cost reduction. At 50% pruning, perplexity roughly doubles and zero-shot accuracy drops by ∼15 points, with 42–45% latency reduction. This aggressive regime is appropriate only for latency-critical applications with high quality tolerance.

5 Conclusion

We presented TriSP (Tri-Signal Structured Pruning), a method that combines activation-weighted magnitude with gradient sensitivity via a geometric mean to produce channel-level importance scores. Paired with adaptive layer-module (AL-AM) allocation and LoRA recovery, TriSP achieves the lowest perplexity and highest downstream accuracy across four models (LLaMA-2-7B, Vicuna-7B, DeepSeek-7B, Mistral-7B) at all tested pruning ratios, while delivering 22–82% throughput improvements through hardware-aligned pruning.

The three ablation studies confirm that each component contributes independently: the TriSP metric outperforms either WIFN or GS alone, AL-AM allocation becomes critical at aggressive pruning (reducing perplexity by 25.83 points at 50%), and LoRA recovery outperforms bias compensation by an increasing margin as pruning intensifies (30.7% vs. 17.8% perplexity reduction at 40%). The systematic comparison between bias compensation and LoRA recovery confirms that input-dependent correction is essential at aggressive compression ratios.

Current limitations include the restriction to decoder-only transformer architectures at 7B scale, and evaluation limited to English-language benchmarks. Future work includes scaling to larger models (30B+), evaluating on broader multilingual benchmarks, and applying TriSP to vision-language models (VLMs).

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Yongqi An, Xu Zhao, Tao Yu, Ming Tang, and Jinqiao Wang. Fluctuation-based adaptive structured pruning for large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 10865–10873, 2024.

Vaswani Ashish. Attention is all you need. *Advances in neural information processing systems*, 30:I, 2017.

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. *arXiv preprint arXiv:2401.15024*, 2024.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *LMSYS blog*, 2023.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Rocktim Jyoti Das, Mingjie Sun, Liqun Ma, and Zhiqiang Shen. Beyond size: How gradients shape pruning decisions in large language models. *arXiv preprint arXiv:2311.04902*, 2023.

Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning*, pp. 10323–10337. PMLR, 2023.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

Babak Hassibi and David Stork. Second order derivatives for network pruning: Optimal brain surgeon. *Advances in neural information processing systems*, 5, 1992.

Yang He and Lingao Xiao. Structured pruning for deep convolutional neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):2900–2919, 2023.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3, 2022.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Steven Kolawole, Lucio Dery, Jean-François Kagy, Virginia Smith, Graham Neubig, and Ameet Talwalkar. Everybody prune now: Structured pruning of llms with only forward passes. *arXiv preprint arXiv:2402.05406*, 2024.

Qi Le, Enmao Diao, Ziyan Wang, Xinran Wang, Jie Ding, Li Yang, and Ali Anwar. Probe pruning: Accelerating llms through dynamic pruning via model probing. *arXiv preprint arXiv:2502.15618*, 2025.

Linfeng Li and Lei Guo. Dynamic low-rank adaptation based pruning algorithm for large language models. In *2024 7th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, pp. 1094–1099. IEEE, 2024.

Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems*, 37:107112–107137, 2024.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in Neural Information Processing Systems*, 36:21702–21720, 2023.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2381–2391, 2018.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Mingjie Sun, Zhuang Liu, Anna Bair, and J. Zico Kolter. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*, 2023.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*, 3(6):7, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Yifan Yang, Kai Zhen, Bhavana Ganesh, Aram Galstyan, Goeric Huybrechts, Markus M"uller, Jonas M. K"ubler, Rupak Vignesh Swaminathan, Athanasios Mouchtaris, Sravan Babu Bodapati, et al. Wanda++: Pruning large language models via regional gradients. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 4321–4333, 2025.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Mingyang Zhang, Hao Chen, Chunhua Shen, Zhen Yang, Linlin Ou, Xinyi Yu, and Bohan Zhuang. Loraprune: Structured pruning meets low-rank parameter-efficient fine-tuning. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3013–3026, 2024.

Longguang Zhong, Fanqi Wan, Ruijun Chen, Xiaojun Quan, and Liangzhi Li. Blockpruner: Fine-grained pruning for large language models. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 5065–5080, 2025.

Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 12:1556–1577, 2024.

A Perplexity Scaling

Figure 4 visualizes the perplexity scaling behavior on LLaMA-2-7B across pruning ratios. Both FLAP and TriSP degrade gracefully at moderate ratios, but diverge sharply at 50%: TriSP + LoRA maintains 13.44 while FLAP + bias rises to 16.03. TriSP without any recovery (27.63) already outperforms FLAP without recovery (29.43).

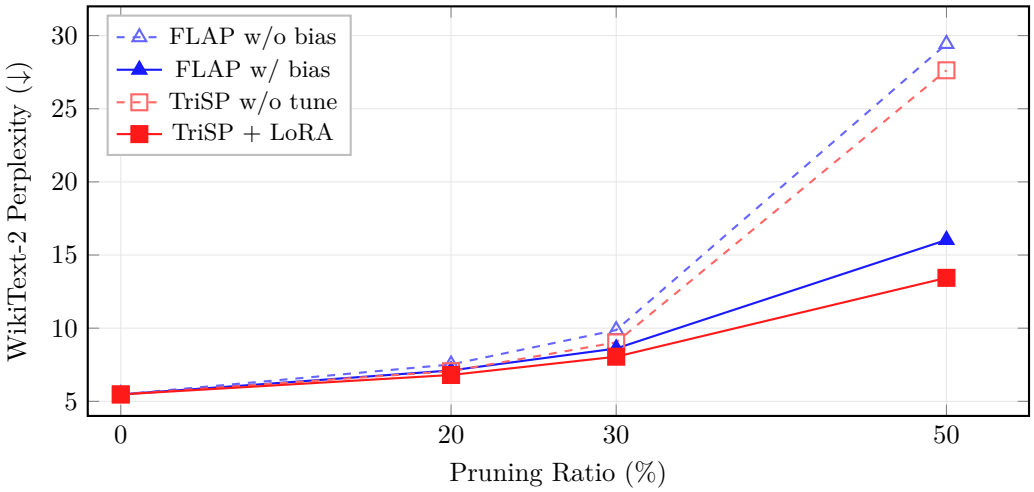

Figure 4: WikiText-2 perplexity scaling with pruning ratio on LLaMA-2-7B. TriSP + LoRA maintains the lowest perplexity at all ratios, with the gap widening significantly at 50% pruning (13.44 vs. 16.03 for FLAP + bias).

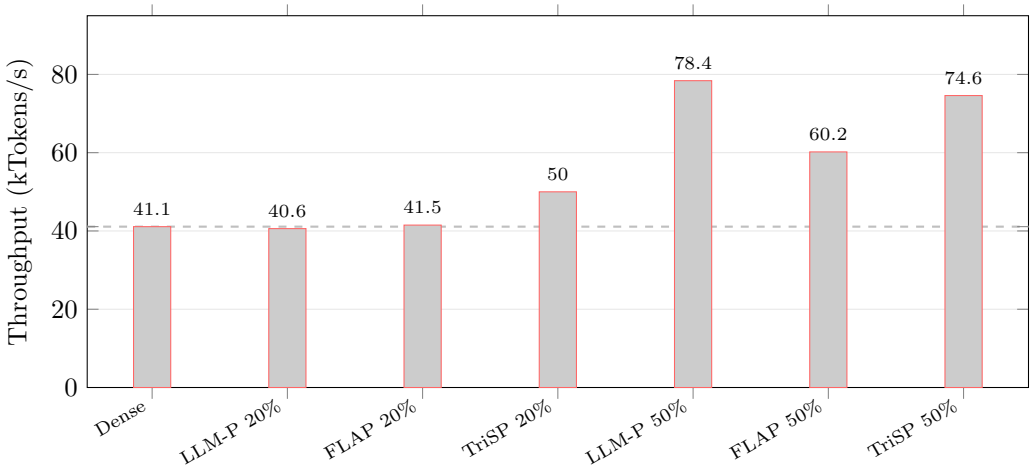

Figure 5: Inference throughput of LLaMA-2-7B (seq=64, batch=32, H100). At 20% pruning, TriSP achieves 50.0k tokens/s (↑22%) while LLM-Pruner and FLAP show no meaningful speedup. At 50%, TriSP delivers 74.6k tokens/s (↑82%) with 45% latency reduction.

B Inference Throughput

Figure 5 reports inference throughput for compressed LLaMA-2-7B models. At 20% pruning, TriSP achieves 50.0k tokens/s (↑22%), while LLM-Pruner and FLAP show no meaningful speedup. At 50%, TriSP delivers 74.6k tokens/s with 45% latency reduction.

C DeepSeek-7B Inference Analysis

Table 8 extends the inference analysis to DeepSeek-7B. TriSP delivers the lowest latency at 20% and 30%. Figure 6 shows that the throughput gap widens with batch size, and Table 9 quantifies the speedup at sequence length 512.

Table 8: Statistics of the compressed DeepSeek-7B model. Inference is conducted with a sequence of 64 tokens at batch size 32 on a single NVIDIA H100.

Method	Ratio	Params	MACs	Memory	Latency (ms)	Tokens/s
DeepSeek-7B	0%	6.91B	415.40G	13181 MiB	47.90	42.8k
LLM-Pruner	20%	5.72B	339.29G	10912 MiB	46.26 (↓3%)	44.3k
FLAP	20%	5.70B	337.75G	10914 MiB	45.13 (↓6%)	45.4k
TriSP (Ours)	20%	5.69B	337.04G	10917 MiB	41.00 (↓14%)	50.0k
LLM-Pruner	30%	5.13B	301.24G	9778 MiB	43.30 (↓10%)	47.3k
FLAP	30%	5.09B	298.93G	9753 MiB	44.15 (↓8%)	46.4k
TriSP (Ours)	30%	5.08B	298.16G	9797 MiB	36.47 (↓24%)	56.2k
LLM-Pruner	50%	3.87B	221.12G	7391 MiB	<u>26.45</u> (↓45%)	77.4k
FLAP	50%	3.88B	221.20G	7442 MiB	35.30 (↓26%)	58.0k
TriSP (Ours)	50%	3.87B	220.50G	7470 MiB	27.97 (↓42%)	73.2k

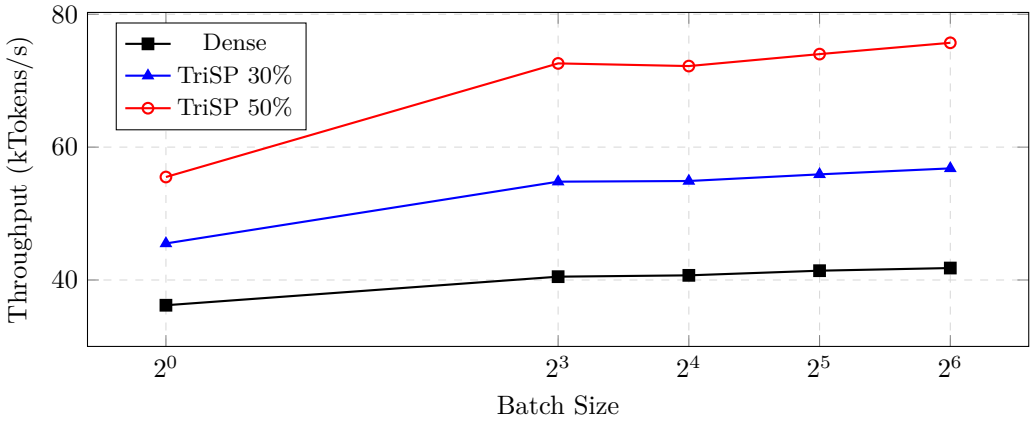

Figure 6: Throughput vs. batch size for TriSP-pruned DeepSeek-7B at sequence length 512. Structured pruning yields consistent throughput gains that scale with batch size in the compute-bound regime.

D Per-Layer Retention under AL-AM

Figure 7 visualizes the per-layer retention ratio under AL-AM allocation for LLaMA-2-7B. At mild pruning ($p = 0.1$), the allocation is nearly uniform. As the ratio increases, a U-shaped pattern emerges: the first and last layers are increasingly preserved while middle layers are pruned most aggressively.

Table 9: Speedup of TriSP-pruned DeepSeek-7B relative to the dense baseline at sequence length 512.

Batch Size	TriSP 30%	TriSP 50%
1	1.26×	1.53×
8	1.35×	1.79×
16	1.35×	1.77×
32	1.35×	1.79×
64	1.36×	1.81×

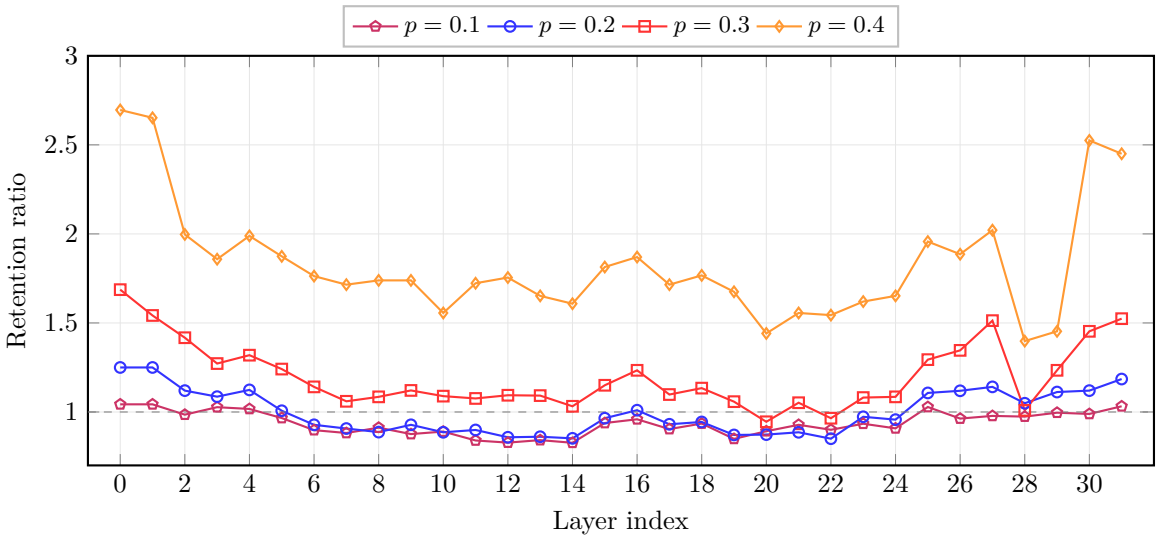

Figure 7: Per-layer retention ratio under AL-AM allocation for LLaMA-2-7B at four pruning ratios. Values above the dashed line (1.0) indicate layers where more parameters are retained relative to a uniform baseline. At mild pruning ($p = 0.1$), the allocation is nearly uniform; as the ratio increases, the U-shaped pattern intensifies, with first and last layers increasingly preserved while middle layers (12–22) are pruned most aggressively.